Genome Read In-Memory (GRIM) Filter

Fast Location Filtering in DNA Read Mapping with Emerging Memory Technologies

Jeremie Kim,

Damla Senol, Hongyi Xin, Donghyuk Lee, Mohammed Alser,
Hasan Hassan, Oguz Ergin, Can Alkan, and Onur Mutlu

Carnegie Mellon

İhsan Doğramacı Bilkent University 1984

TOBB
UNIVERSITY OF
ECONOMICS AND TECHNOLOGY

SAFARI

Introduction

- 3D-stacked Memory: an emerging technology
  - Processing-in-Memory (PIM) allows embedding customized logic
  - Enables high bandwidth

- Read mapping can utilize this technology to gain major performance improvements because it is:
  - Compute intensive
  - Memory intensive

- **Goal**: We propose an implementation of read mapping using Processing-in-Memory (PIM) for acceleration

Hash Table Based Read Mappers

- Our work focuses on hash table based read mappers

- The filtering step in read mappers is now the bottleneck
- Mappers align billions of reads, most incorrect mappings
- **Filter Purpose:** quickly rejects incorrect mappings before alignment – to reduces costly edit distance calculations
- **Costly because**: they are compute and memory intensive
  - Called for every candidate mapping location
  - Filtering each location requires nontrivial compute / multiple memory accesses

- How can we alleviate the bottleneck?

Problem

- Filters are generally either fast or accurate, i.e.
  - FastHASH **[Xin+, BMC Genomics 2013]**
    - Fast but inaccurate under high error tolerance settings
  - Q-Gram **[Rasmussen+, Journal of Computational Biology 2006]**
    - Slow but accurate

- **We Propose**:
  - GRIM-Filter
    - Faster than FastHASH with the accuracy of q-gram
    - Accomplished this by employing an emerging memory technology

Key Ideas

- GRIM-Filter, a PIM-friendly filtering algorithm that is both fast and accurate.

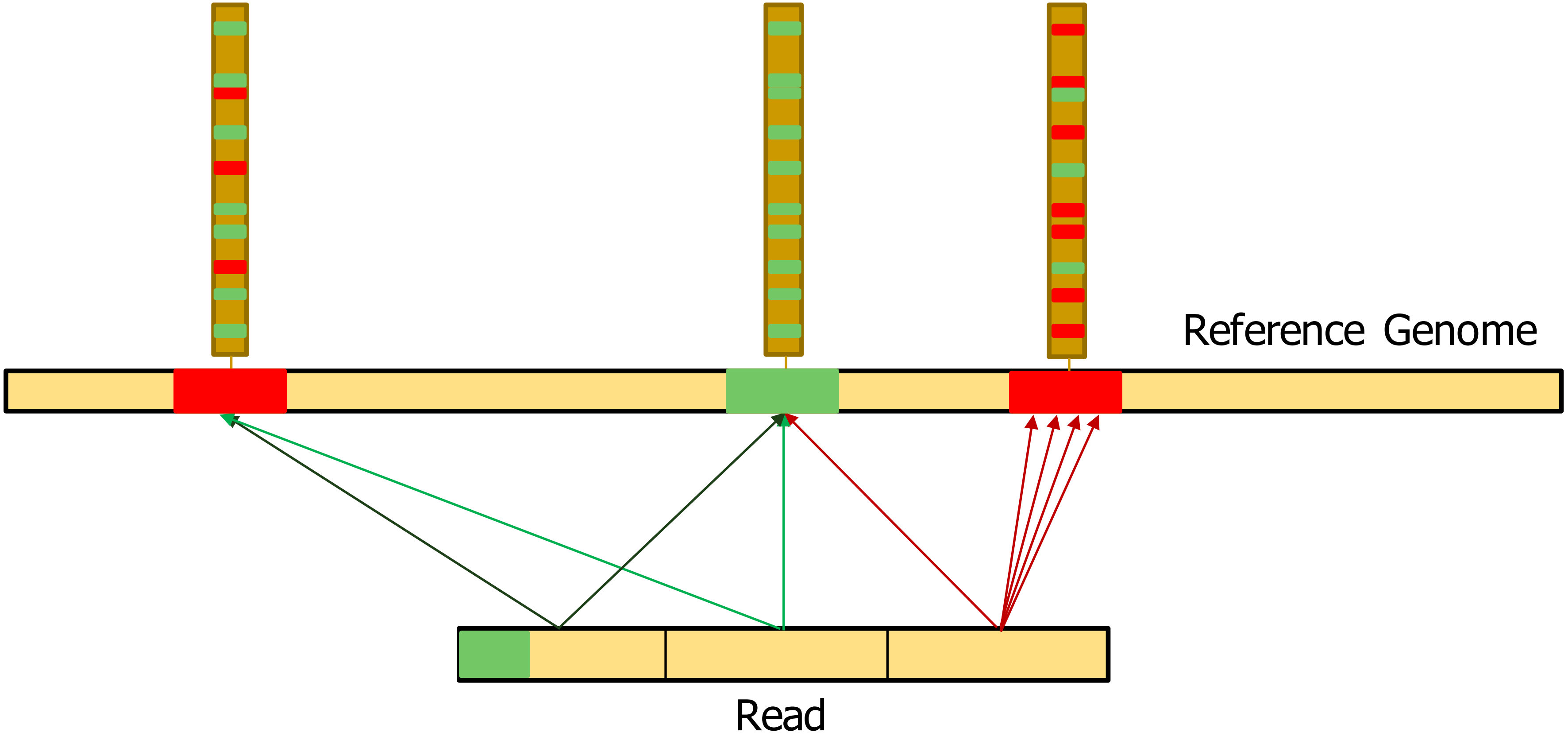
- GRIM-Filter is built upon two key ideas

1. Modify q-gram string matching
   - Enables concurrent checking for multiple locations

2. Utilize a 3D-stacked DRAM architecture
   - Alleviates memory bandwidth issue
   - Parallelizes most of the filter

Key Idea 1 – Q-gram Modification

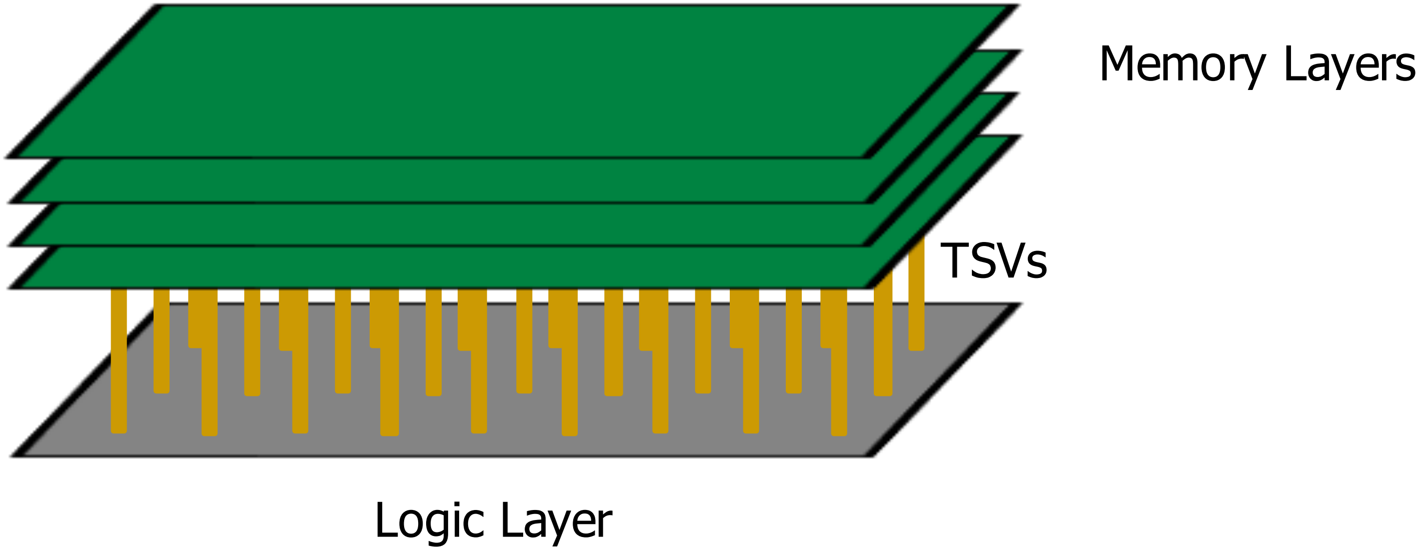
Modify q-gram string matching for concurrently checking for multiple locations.



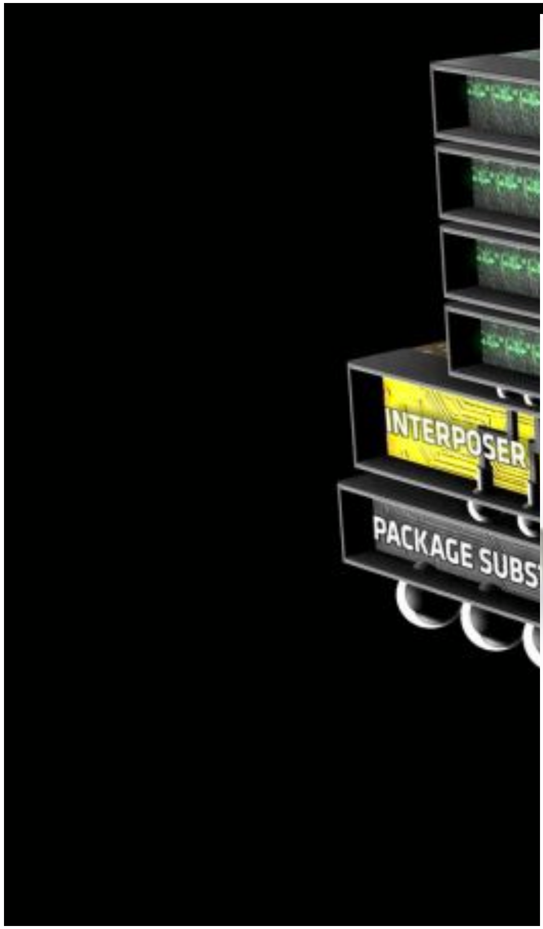
Key Idea 2 – Utilize 3D-stacked Memory

- 3D-stacked DRAM architecture is extremely high bandwidth and can parallelize most of the filter

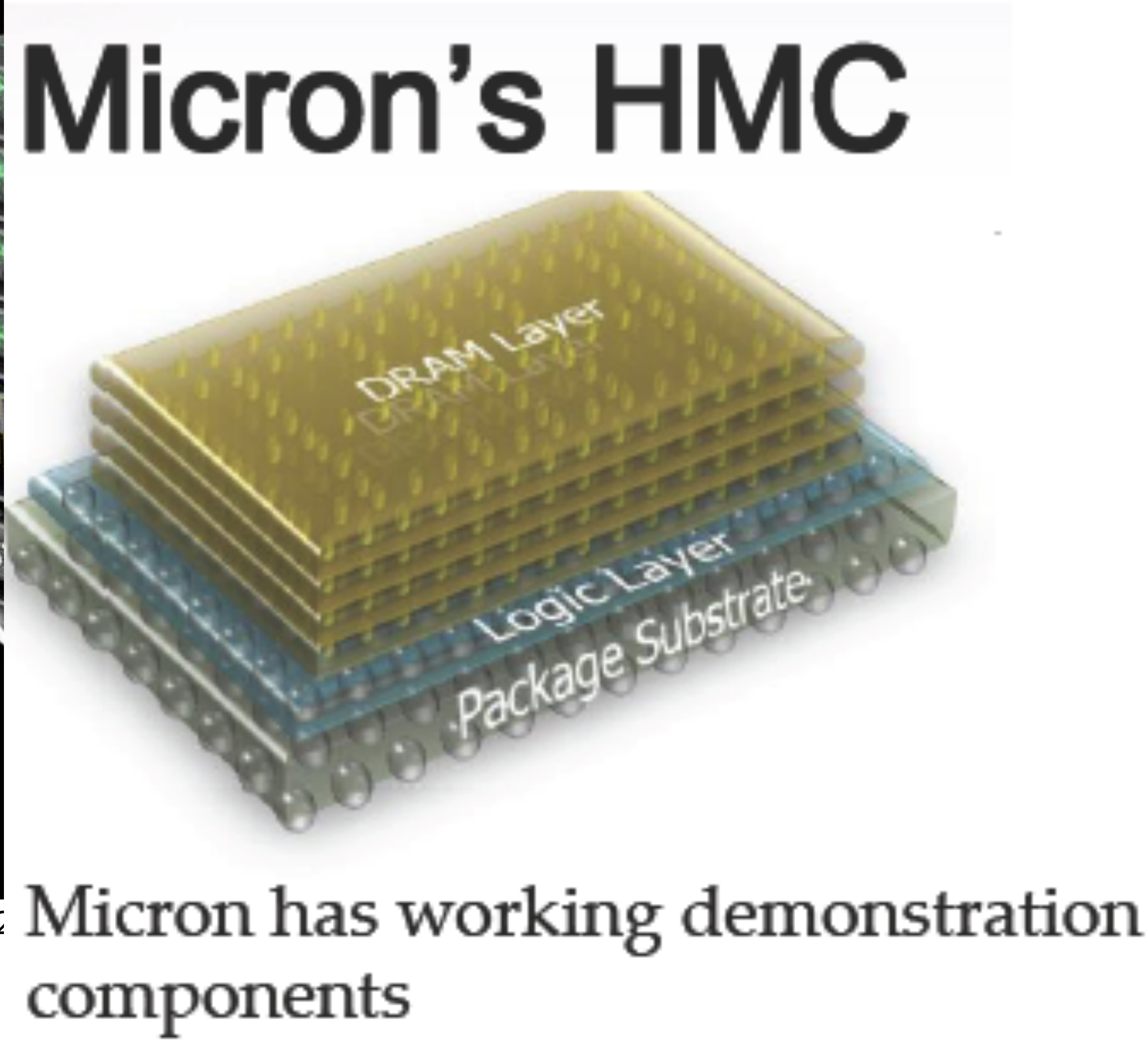
- Embed GRIM-Filter into DRAM logic layer and appropriately distribute bitvectors throughout memory



Key Idea 2 – Utilize 3D-stacked Memory



http://images.anandtech.com/doci/92

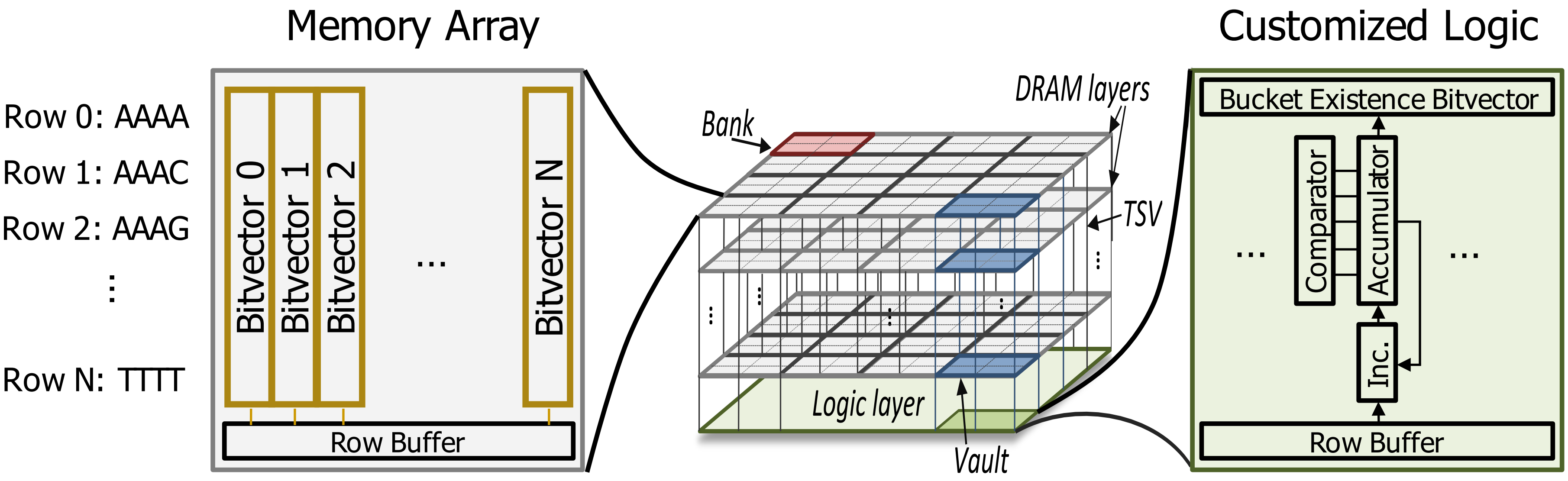

Micron's HMC

Micron has working demonstration components

http://i1-news.softpedia-static.com/images/news2/Micron-and-Samsung-Join-Force-to-Create-Next-Gen-Hybrid-Memory-2.png

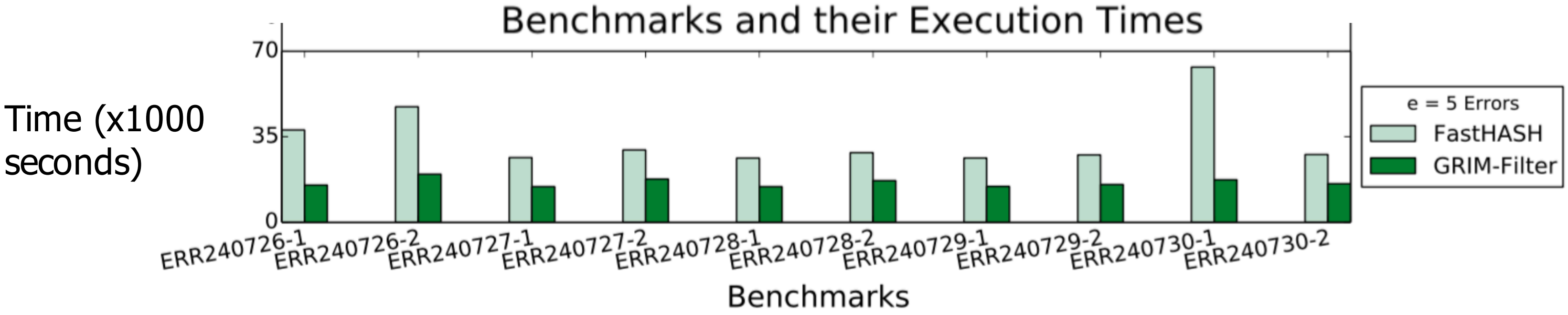
Q-gram Modified in 3D stacked DRAM

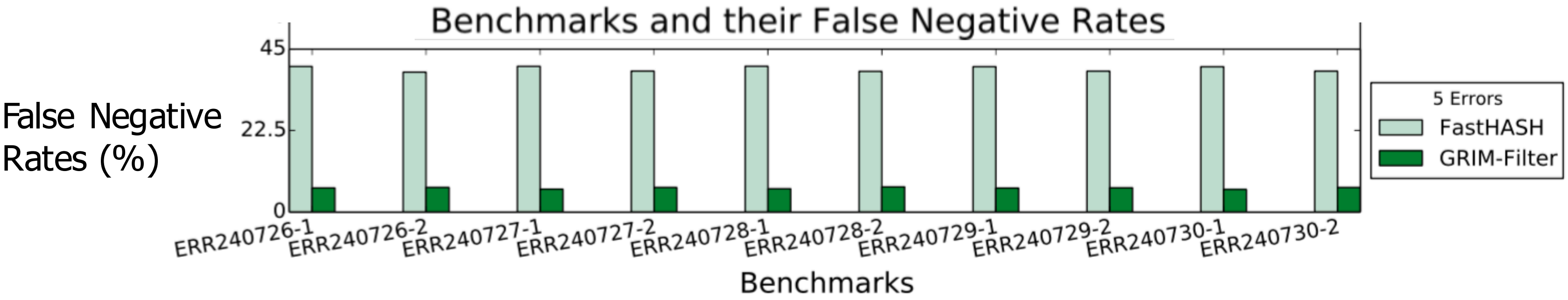
- We employ both key ideas to implement the following figure to modify q-gram filtering in order to make it more amenable for processing-in-memory



Key Results

2.08x average performance benefit on real data sets





5.97x reduction in False Negative Rate on real data sets

Conclusions

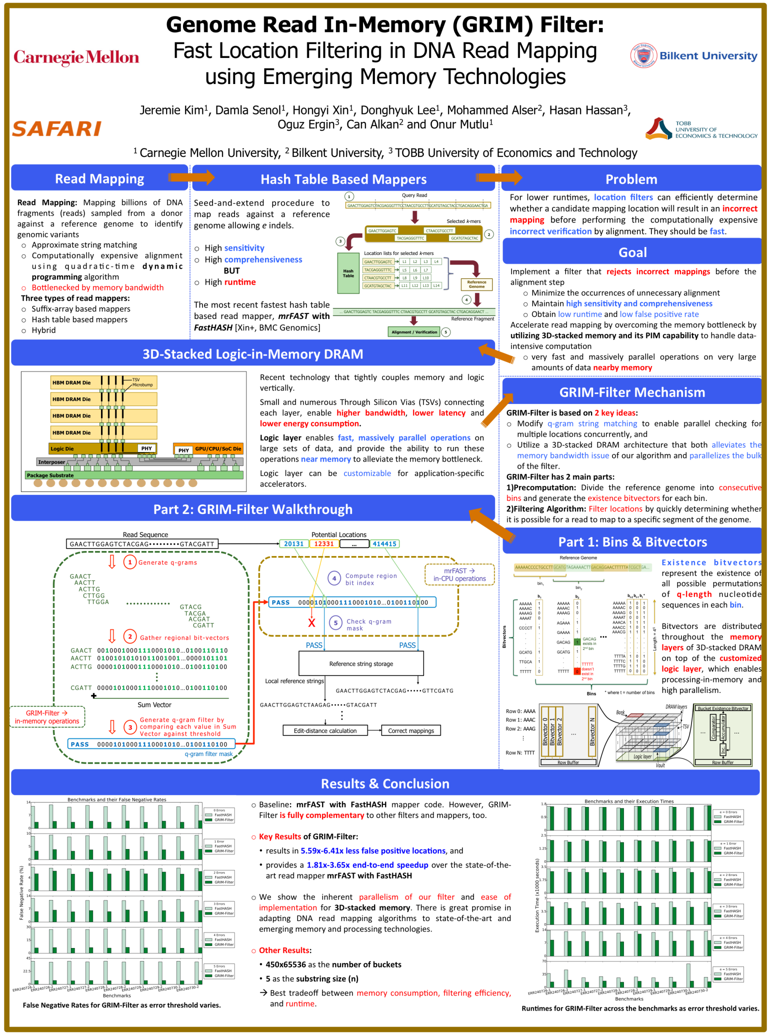
- We propose an in memory filter that can drastically speed up read mapping

- Compared to the previous best filter
  - We observed 1.81x-3.65x speedup
  - We observed 5.59x-6.41x fewer false negatives

- GRIM-Filter is a universal filter that can be applied to any read mapper

Thank You!

Poster #118



Genome Read In-Memory (GRIM) Filter: Fast Location Filtering in DNA Read Mapping using Emerging Memory Technologies

Jeremie Kim[1], Damla Senol[1], Hongyi Xin[1], Donghyuk Lee[1], Mohammed Alser[2], Hasan Hassan[3], Oguz Ergin[3], Can Alkan[2] and Onur Mutlu[1]

[1] Carnegie Mellon University, [2] Bilkent University, [3] TOBB University of Economics and Technology

Read Mapping

Read Mapping: Mapping billions of DNA fragments (reads) sampled from a donor against a reference genome to identify genomic variants

- Approximate string matching
- Computationally expensive alignment using quadratic-time **dynamic programming** algorithm
- Bottlenecked by memory bandwidth

Three types of read mappers:

- Suffix-array based mappers
- Hash table based mappers
- Hybrid

Hash Table Based Mappers

Seed-and-extend procedure to map reads against a reference genome allowing *e* indels.

- High sensitivity
- High comprehensiveness

  BUT
- High runtime

The most recent fastest hash table based read mapper, ***mrFAST* with *FastHASH*** [Xin+, BMC Genomics]

1 Query Read
GAACTTGGAGTCTACGAGGGTTTCCTAACGTGCCTTGCATGTAGCTACCTGACAGGAACTGA
Selected k-mers 2
GAACTTGGAGTC CTAACGTGCCTT TACGAGGGTTTC GCATGTAGCTAC
3
Hash Table
Location lists for selected k-mers
L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12 L13 L14
Reference Genome
4
... GAACTTGGAGTC TACGAGGGTTTC CTAACGTGCCTT GCATGTAGCTAC CTGACAGGAACT ...
Reference Fragment
Alignment / Verification 5

Problem

For lower runtimes, location filters can efficiently determine whether a candidate mapping location will result in an **incorrect mapping** before performing the computationally expensive incorrect verification by alignment. They should be fast.

Goal

Implement a filter that **rejects incorrect mappings** before the alignment step

- Minimize the occurrences of unnecessary alignment
- Maintain **high sensitivity and comprehensiveness**
- Obtain low runtime and low false positive rate

Accelerate read mapping by overcoming the memory bottleneck by **utilizing 3D-stacked memory and its PIM capability** to handle data-intensive computation

- very fast and massively parallel operations on very large amounts of data **nearby memory**

3D-Stacked Logic-in-Memory DRAM

Recent technology that tightly couples memory and logic vertically.

Small and numerous Through Silicon Vias (TSVs) connecting each layer, enable **higher bandwidth**, **lower latency** and **lower energy consumption.**

Logic layer enables fast, massively parallel operations on large sets of data, and provide the ability to run these operations near memory to alleviate the memory bottleneck.

Logic layer can be customizable for application-specific accelerators.

GRIM-Filter Mechanism

GRIM-Filter is based on 2 key ideas:

- Modify q-gram string matching to enable parallel checking for multiple locations concurrently, and
- Utilize a 3D-stacked DRAM architecture that both alleviates the memory bandwidth issue of our algorithm and parallelizes the bulk of the filter.

GRIM-Filter has 2 main parts:

1)Precomputation: Divide the reference genome into consecutive bins and generate the existence bitvectors for each bin.

2)Filtering Algorithm: Filter locations by quickly determining whether it is possible for a read to map to a specific segment of the genome.

Part 2: GRIM-Filter Walkthrough

Read Sequence
GAACTTGGAGTCTACGAG•••••••••GTACGATT
Potential Locations
20131 12331 ... 414415
1 Generate q-grams
GAACT AACTT ACTTG CTTGG TTGGA GTACG TACGA ACGAT CGATT
2 Gather regional bit-vectors
GAACT 001000100011100010 10...0100110110
AACTT 0100101010101100100 1...0000101101
ACTTG 0000101000111000101 0...0100110100
CGATT 0000101000111000101 0...0100110100
Sum Vector
GRIM-Filter → in-memory operations
3 Generate q-gram filter by comparing each value in Sum Vector against threshold
PASS 0000101000111000101 0...0100110100
q-gram filter mask
4 Compute region bit index
mrFAST → in-CPU operations
PASS 0000101000111000101 0...0100110100
X
5 Check q-gram mask
PASS PASS
Reference string storage
Local reference strings
GAACTTGGAGTCTACGAG•••••GTTCGATG
GAACTTGGAGTCTAAGAG•••••GTACGATT
Edit-distance calculation
Correct mappings

Part 1: Bins & Bitvectors

Reference Genome
AAAAACCCCTGCCTTGCATGTAGAAAACTTGACAGGAACTTTTTATCGCTGA...
bin1 bin2
Bitvectors
b1 b2 bt-2 bt-1 bt*
GACAG exists in 2nd bin
TTTTT doesn't exist in 2nd bin
Length = 4^5
Bins
* where t = number of bins
Row 0: AAAA
Row 1: AAAC
Row 2: AAAG
Row N: TTTT
Bitvector 0 Bitvector 1 Bitvector 2 Bitvector N
Row Buffer
Bank
DRAM layers
TSV
Logic layer
Vault
Bucket Existence Bitvector
Comparator Accumulator Inc.

Existence bitvectors represent the existence of all possible permutations of **q-length** nucleotide sequences in each bin.

Bitvectors are distributed throughout the **memory layers** of 3D-stacked DRAM on top of the **customized logic layer**, which enables processing-in-memory and high parallelism.

Results & Conclusion

Benchmarks and their False Negative Rates
False Negative Rate (%)
0 Errors 1 Error 2 Errors 3 Errors 4 Errors 5 Errors
FastHASH
GRIM-Filter
Benchmarks

False Negative Rates for GRIM-Filter as error threshold varies.

- Baseline: **mrFAST with FastHASH** mapper code. However, GRIM-Filter **is fully complementary** to other filters and mappers, too.
- **Key Results** of GRIM-Filter:
  - results in **5.59x-6.41x less false positive locations**, and
  - provides a **1.81x-3.65x end-to-end speedup** over the state-of-the-art read mapper **mrFAST with FastHASH**
- We show the inherent parallelism of our filter and ease of implementation for **3D-stacked memory**. There is great promise in adapting DNA read mapping algorithms to state-of-the-art and emerging memory and processing technologies.
- **Other Results:**
  - **450x65536** as the **number of buckets**
  - **5** as the **substring size (n)**

  → Best tradeoff between memory consumption, filtering efficiency, and runtime.

Runtimes for GRIM-Filter across the benchmarks as error threshold varies.

Genome Read In-Memory (GRIM) Filter

Fast Location Filtering in DNA Read Mapping with Emerging Memory Technologies

Jeremie Kim,
Damla Senol, Hongyi Xin, Donghyuk Lee, Mohammed Alser,
Hasan Hassan, Oguz Ergin, Can Alkan, and Onur Mutlu

Carnegie Mellon

İhsan Doğramacı Bilkent University
Bilkent
1984

TOBB
UNIVERSITY OF
ECONOMICS AND TECHNOLOGY

SAFARI